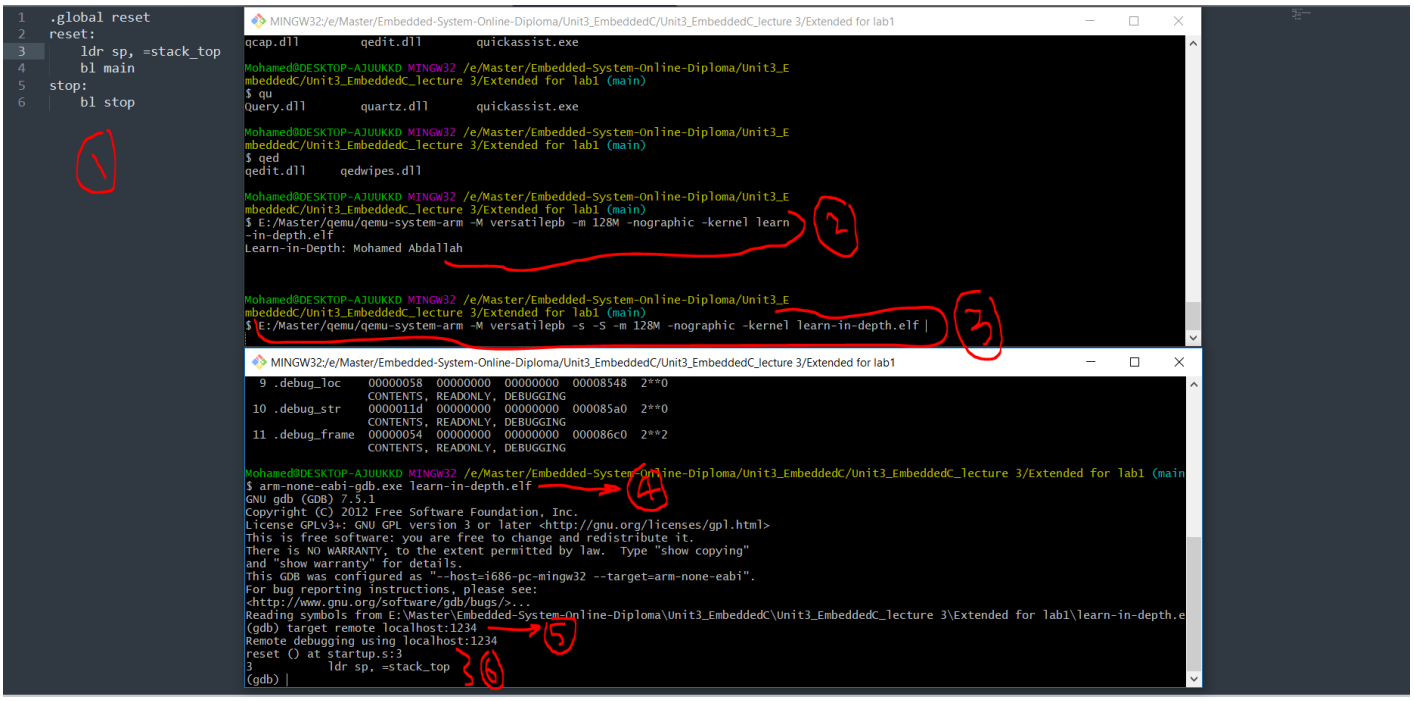
REPORT
FOR
EXTENDED
LAB1



1. Startup.s file (assembly).
2. Turning on our board (versatilepb) and burning our barmetal software application (writing on uart Learn-in-Depth: Mohamed Abdallah).
3. My board is ready and wait debugger.
4. Path learn-in-depth.elf to debug (not .bin as it's not contain debug info).
5. Open path with my board (localhost → my machine IP, 1234 → port, info from board datasheet).
6. Processor @ entry point (reset section at startup file loading value of stack_top to stack pointer "initializing stack").

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Master\Embedded-System-Online-Diploma\Unit3_EmbeddedC\Un
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3           ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3           ldr sp, =stack_top
4           bl main
5       stop:
6           bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      bl      0x10008 <stop>
(gdb) |
```

1. (l) show currently file.
2. (display/3i $pc) show three instructions, currently program counter point to next instruction should be fetched (0x10000 initializing stack then fetch next instruction which will branch to main).

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Master\Embedded-System-Online-Diploma\Unit3_EmbeddedC\Un
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3           ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3           ldr sp, =stack_top
4           bl main
5       stop:
6           bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      bl      0x10008 <stop>
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 9.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 6.
(gdb) |
```

1. (b main) breakpoint at main function (0x10018 first line of c code in main between 0x10010 & 0x10018 context switching and initializing stack of main).
2. (b *0x10010) breakpoint at address 0x10010 to watch context and initializing of main stack.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3           ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3           ldr sp, =stack_top
4           bl main
5       stop:
6           bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      bl      0x10008 <stop>
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 9.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 6.
(gdb) si
reset () at startup.s:4
4           bl main
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      bl      0x10008 <stop>
   0x1000c <stop+4>:    ldrdeq  r1, [r1], -r12
(gdb)
```

1. (si → step instruction) pc will point to next address 0x10004 instruction which will be fetched.
   (s → step c code level one line code in c maybe equivalent to many assembly instruction).

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      bl      0x10008 <stop>
   0x1000c <stop+4>:    ldrdeq  r1, [r1], -r12
(gdb) c
Continuing.

Breakpoint 2, main () at app.c:6
6       {
1: x/3i $pc
=> 0x10010 <main>:      push    {r11, lr}
   0x10014 <main+4>:    add     r11, sp, #4
   0x10018 <main+8>:    ldr     r0, [pc, #4]    ; 0x10024 <main+20>
(gdb) c
Continuing.

Breakpoint 1, main () at app.c:9
9           uart_send_string(stringBuffer);
1: x/3i $pc
=> 0x10018 <main+8>:    ldr     r0, [pc, #4]    ; 0x10024 <main+20>
   0x1001c <main+12>:   bl      0x10028 <uart_send_string>
   0x10020 <main+16>:   pop     {r11, pc}
(gdb) l
4       uint8 stringBuffer[100] = "Learn-in-Depth: Mohamed Abdallah";
5       void main(void)
6       {
7           /* pass string to uart */
8           // name of array point to the address of first charater stringBuffer == &stringBuffer[0]
9           uart_send_string(stringBuffer);
10      }(gdb) |
```

1. (c → continue) will jump to first breakpoint at address 0x10010.
2. (c) again will jump to next breakpoint at address 0x10018 which first c code of main function at line 9.
3. (l show currently file app.c) we found that first line of c main at line 9 already.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) print stringBuffer
$1 = "Learn-in-Depth: Mohamed Abdallah", '\000' <repeats 67 times>
(gdb) print stringBuffer[0]
(gdb) Undefined command: "".  Try "help".
help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) print stringBuffer[0]
$2 = 76 'L'
(gdb) |
```

1. (print stringBuffer[0]) print value of variable 76 asci code.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) display/3i $pc
2: x/3i $pc
=> 0x10038 <uart_send_string+16>:       b       0x10058 <uart_send_string+48>
   0x1003c <uart_send_string+20>:
    ldr r3, [pc, #48]   ; 0x10074 <uart_send_string+76>
   0x10040 <uart_send_string+24>:       ldr     r2, [r11, #-8]
(gdb) where
#0  uart_send_string (
    p_tx_string=0x10078 <stringBuffer> "Learn-in-Depth: Mohamed Abdallah")
    at uart.c:9
#1  0x00010020 in main () at app.c:9
(gdb) |
```

1

- I set breakpoint at uart_send_string then pressed c to jump to it.

1. (where) currently at uart_send_string function in uart.c file and reached here by call was in main function in file app.c at line 9.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -m 128M -s -S -nographic -kernel learn-in-depth.elf
Learn-in-Depth: Mohamed Abdallah

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (ma
in)
$ ^C

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (ma
in)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -m 128M -s -S -nographic -kernel learn-in-depth.elf
L|
```

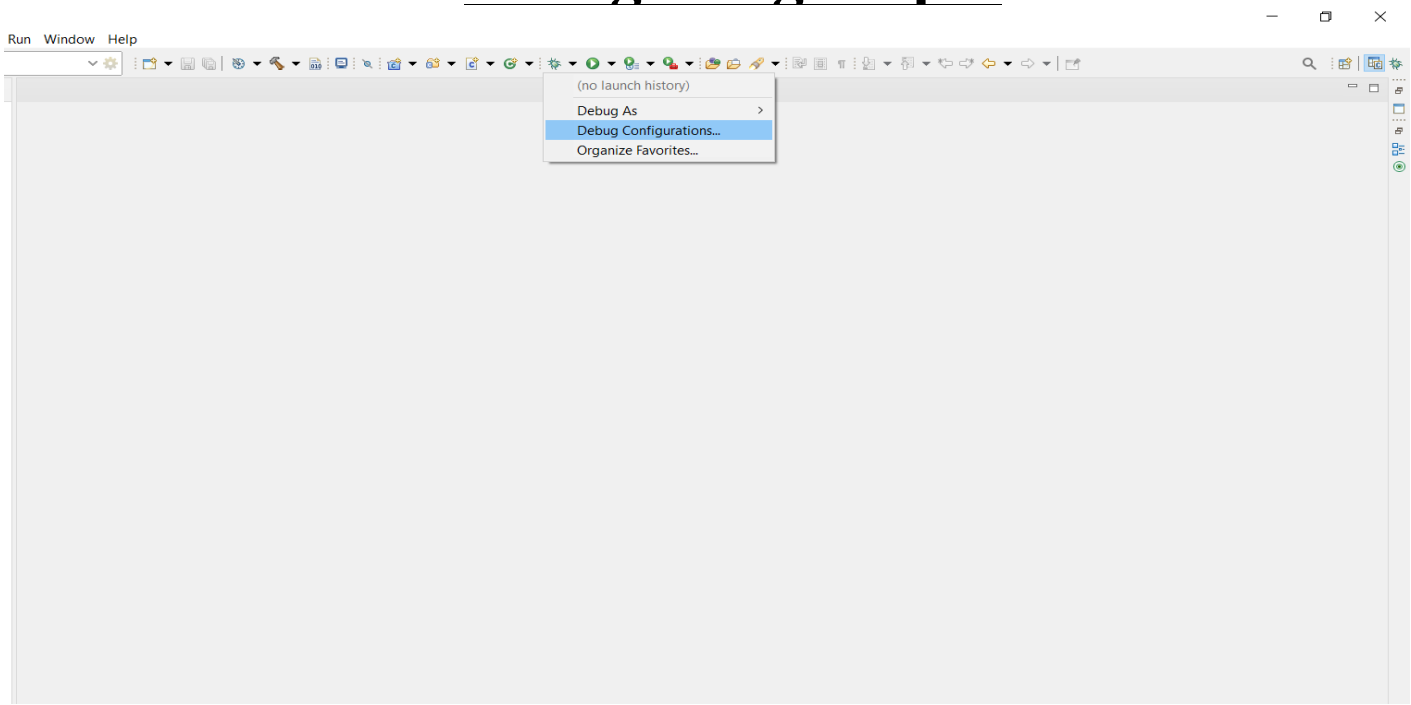
2

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
#1  0x00010020 in main () at app.c:9
(gdb) l
14              p_tx_string++;
15          }
16      }(gdb) s
12              UART0DR = (uint32)*p_tx_string;
2: x/3i $pc
=> 0x1003c <uart_send_string+20>:
    ldr r3, [pc, #48]   ; 0x10074 <uart_send_string+76>
   0x10040 <uart_send_string+24>:       ldr     r2, [r11, #-8]
   0x10044 <uart_send_string+28>:       ldrb    r2, [r2]
1: x/3i $pc
=> 0x1003c <uart_send_string+20>:
    ldr r3, [pc, #48]   ; 0x10074 <uart_send_string+76>
   0x10040 <uart_send_string+24>:       ldr     r2, [r11, #-8]
   0x10044 <uart_send_string+28>:       ldrb    r2, [r2]
(gdb) s
14              p_tx_string++;
2: x/3i $pc
=> 0x1004c <uart_send_string+36>:       ldr     r3, [r11, #-8]
   0x10050 <uart_send_string+40>:       add     r3, r3, #1
   0x10054 <uart_send_string+44>:       str     r3, [r11, #-8]
1: x/3i $pc
=> 0x1004c <uart_send_string+36>:       ldr     r3, [r11, #-8]
   0x10050 <uart_send_string+40>:       add     r3, r3, #1
   0x10054 <uart_send_string+44>:       str     r3, [r11, #-8]
(gdb) |
```

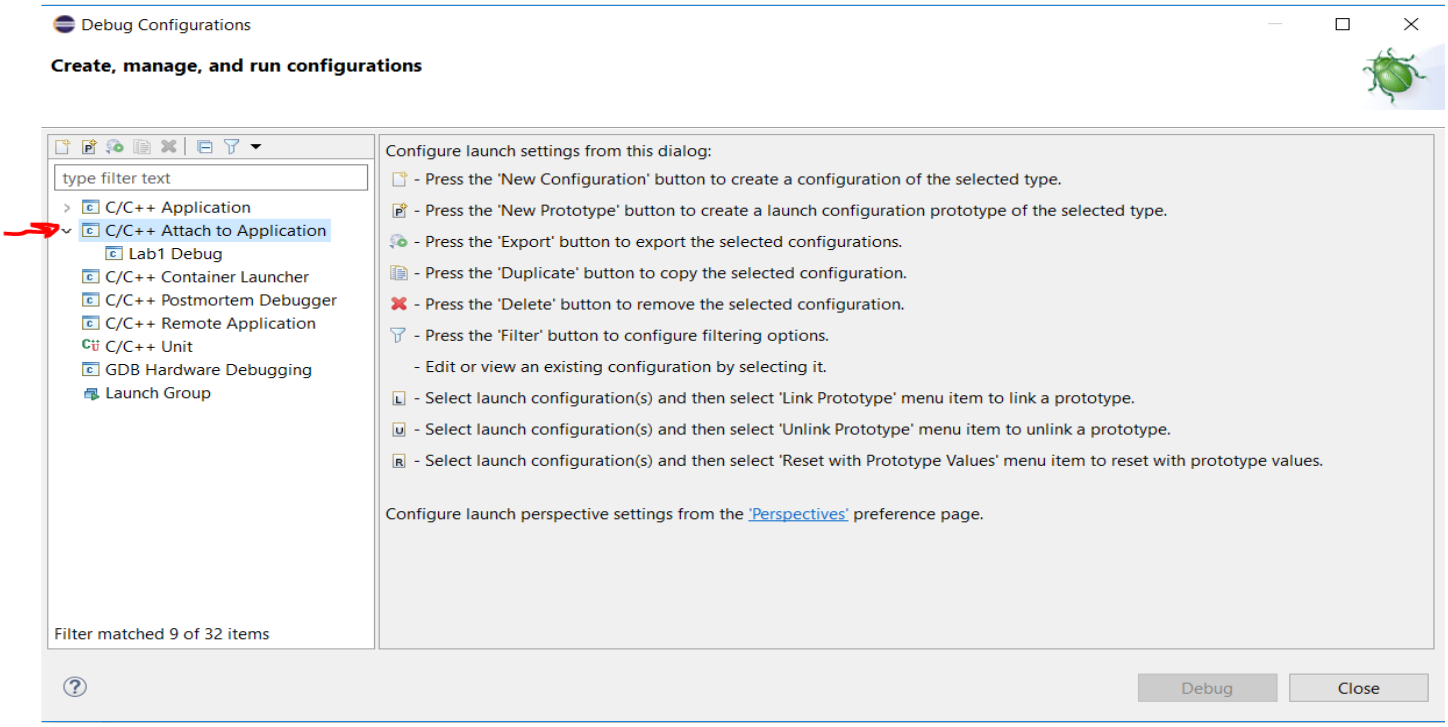
1

1. Using s will move line by line in c code to watch uart displaying our string written character by character.

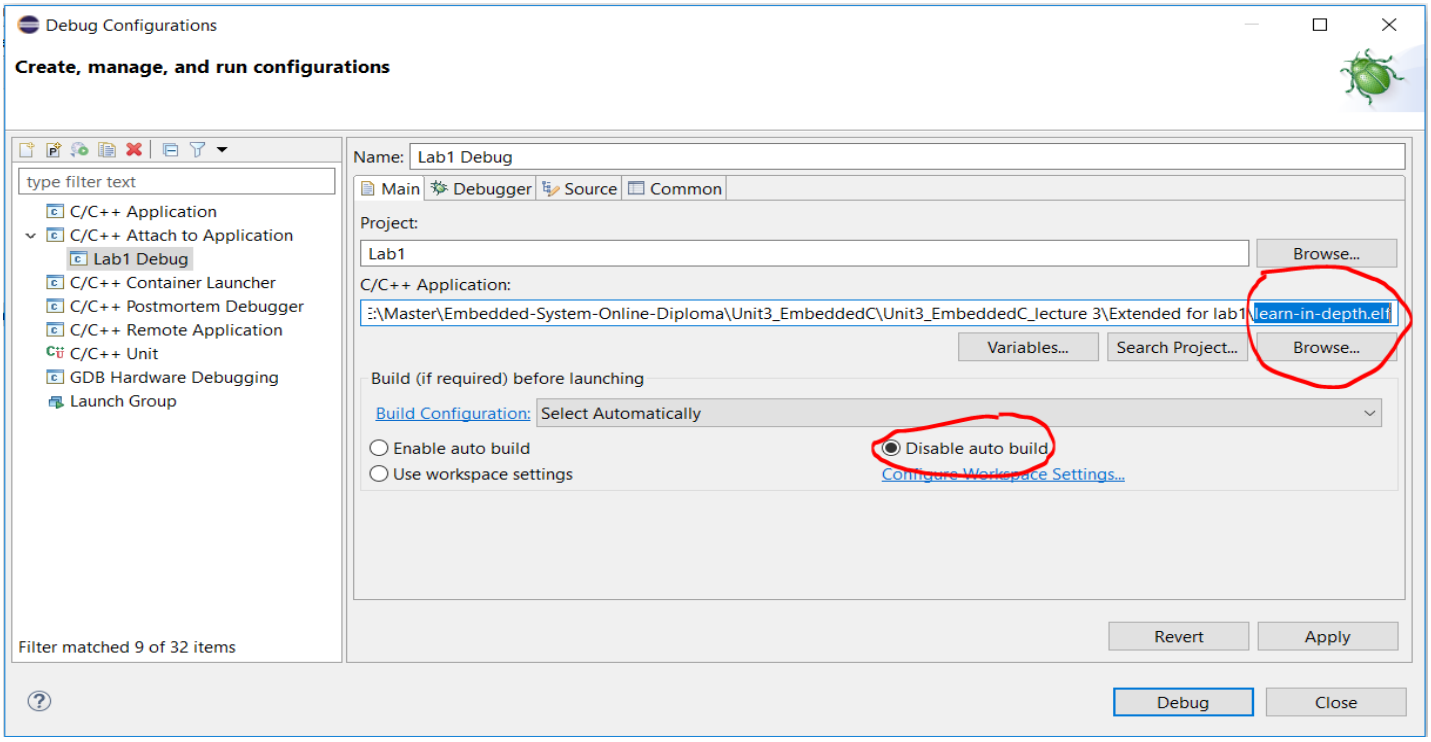
Debug using eclipse



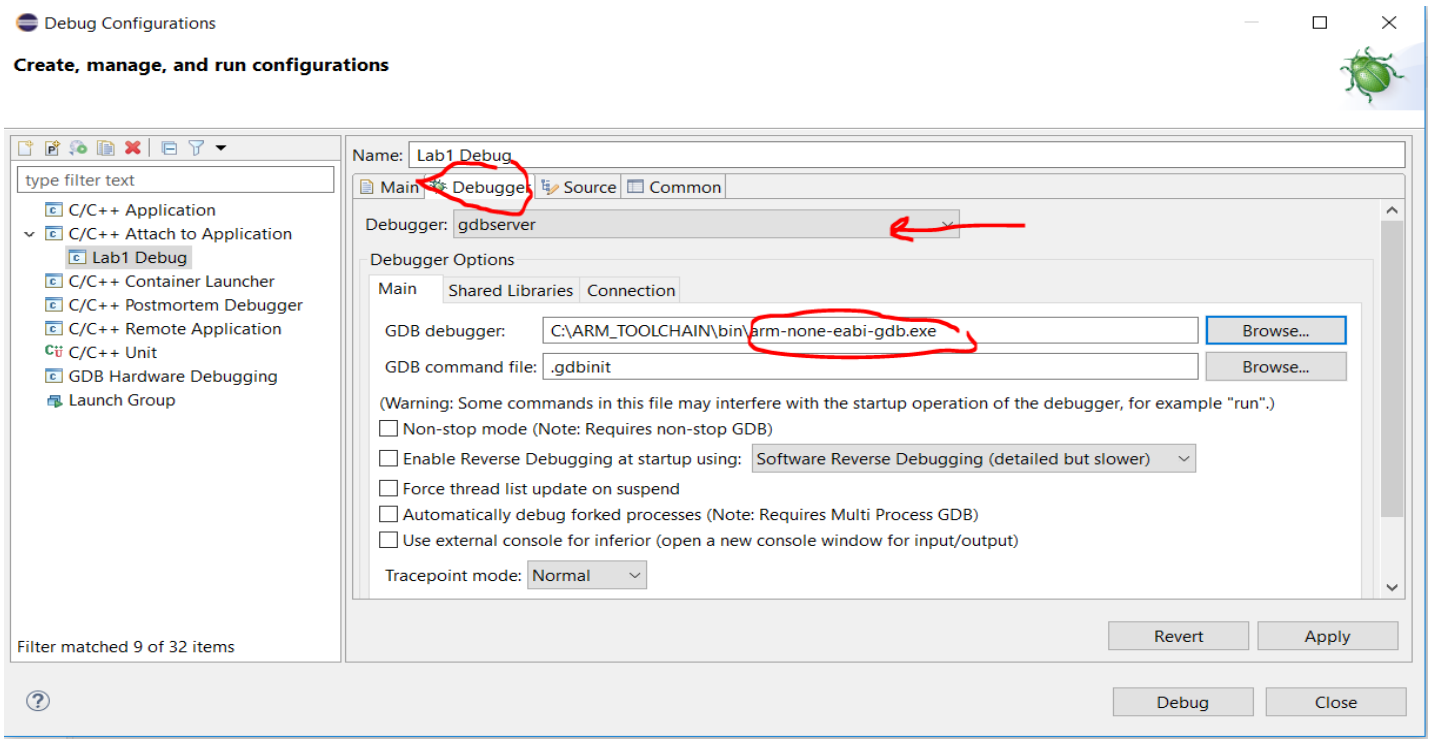
- Open empty project then select debug config.



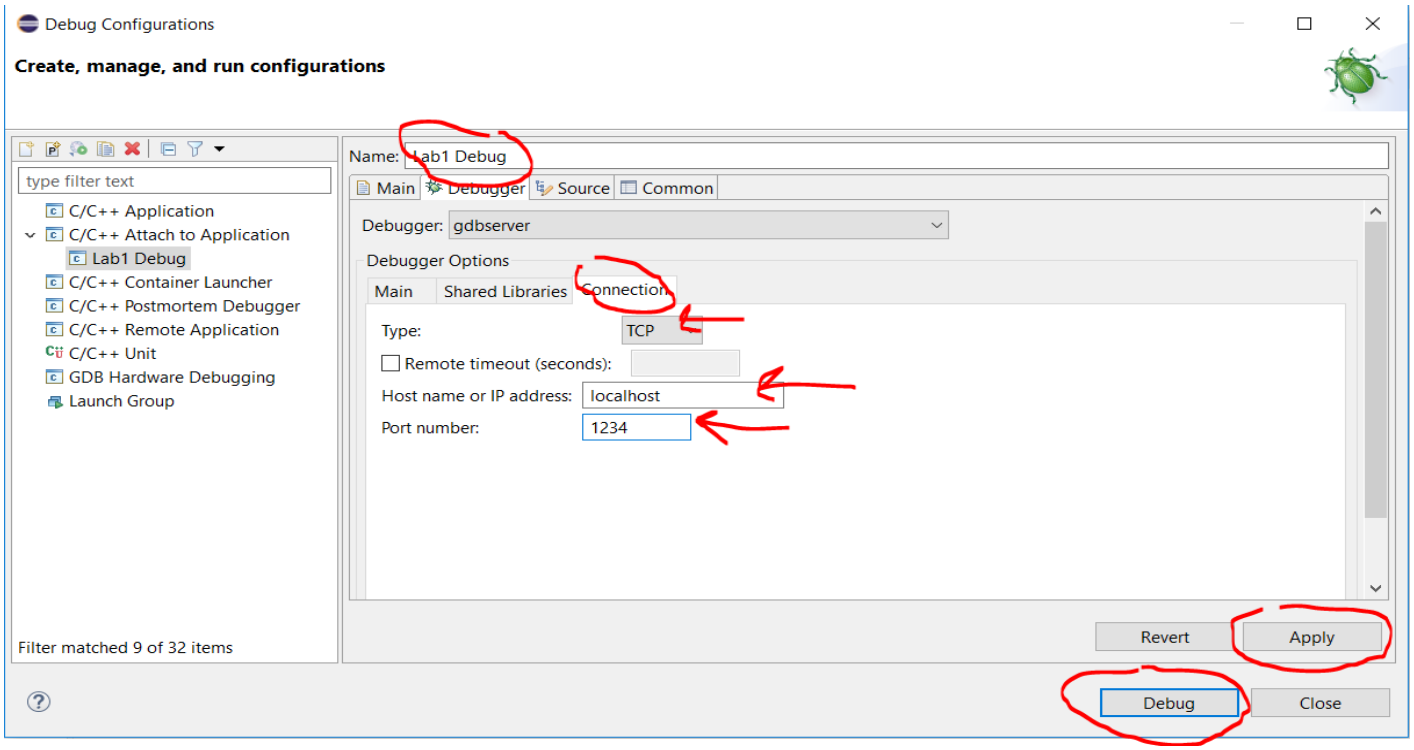
- Then double click on attach to application.



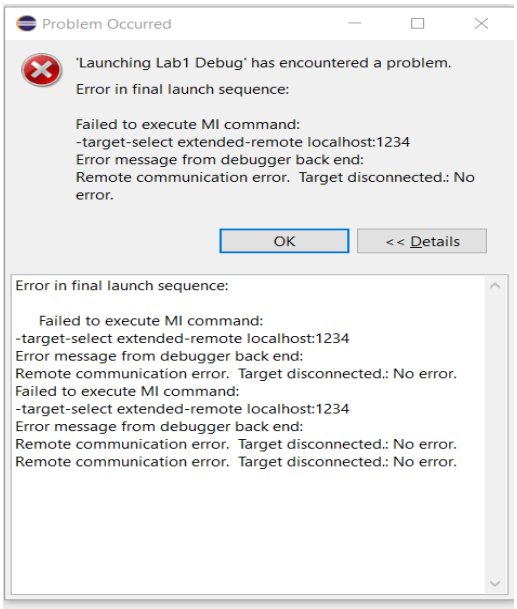
- Then select my .elf file and disable auto bulid.



- Then select debugger tab, select gdbserver then select gdb cross tool chain not native as our code will not run on our pc it will run on anther machine.



- Select connection tab and select Type and set IP and port number.
- Give name to that debug config then applay and Debug.



- Maybe face this problem if you forgot to Open your board.

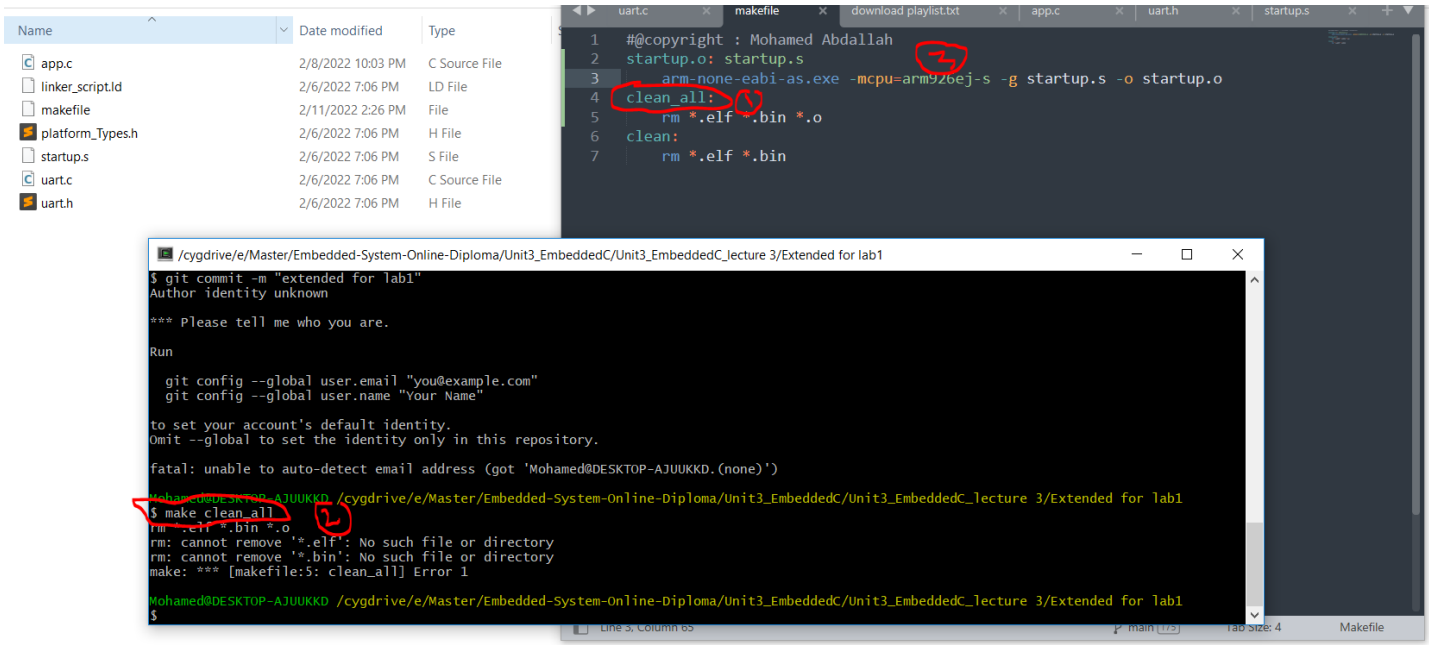
```
Console  Registers  Problems  Executables  Debugger Console ×  Memory
Lab1 Debug [C/C++ Attach to Application] C:\ARM_TOOLCHAIN\bin\arm-none-eabi-gdb.exe (7.5.1)
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
pwd
Working directory E:\Master\Embedded-System-Online-Diploma\unit2_C_Programming\codeforces\Lab1.
file learn-in-depth.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) [answered Y; input not from terminal]
Reading symbols from E:\Master\Embedded-System-Online-Diploma\unit2_C_Programming\codeforces\Lab1\learn-in-depth.elf...done.
```

- You need to know your current directory by using pwd and copy your .elf there then write this command file file-name.elf.
- You will see that gdb can now read symbols successfully.

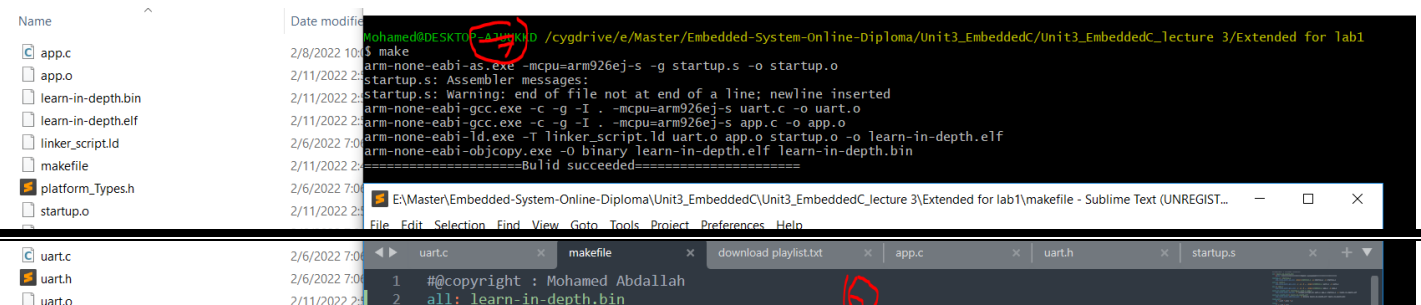
```
Debug ×  Project Explorer
Lab1 Debug [C/C++ Attach to Applicatior
  learn-in-depth.elf
    Thread #1 1 (CPU#0 [running]) (Susp
      reset() at startup.s:4 0x10004
  C:\ARM_TOOLCHAIN\bin\arm-none-ea

0x10000  startup.s ×
1 .global reset
2 reset:
3     ldr sp, =stack_top
4     bl main
5 stop:
6     bl stop

Variables  Breakpo...  Expressi...  Modules  Disasse... ×
Enter location here
0000fff0:   andeq   r0, r0, r0
0000fff4:   andeq   r0, r0, r0
0000fff8:   andeq   r0, r0, r0
0000fffc:   andeq   r0, r0, r0
00010000:   ldr     sp, [pc, #4]    ; 0x1000c
00010004:   bl      0x10010
00010008:   bl      0x10008
0001000c:   ldrdeq  r1, [r1], -r12
00010010:   push    {r11, lr}
00010014:   add     r11, sp, #4
00010018:   ldr     r0, [pc, #4]    ; 0x10024
0001001c:   bl      0x10028
00010020:   pop     {r11, pc}
00010024:   andeq   r0, r1, r8, ror r0
00010028:   push    {r11}           ; (str r11, [sp, #-
0001002c:   add     r11, sp, #0
00010030:   sub     sp, sp, #12
00010034:   str     r0, [r11, #-8]
00010038:   b       0x10058
0001003c:   ldr     r3, [pc, #48]   ; 0x10074
00010040:   ldr     r2, [r11, #-8]
00010044:   ldrb    r2, [r2]
00010048:   str     r2, [r3]
0001004c:   ldr     r3, [r11, #-8]
00010050:   add     r3, r3, #1
00010054:   str     r3, [r11, #-8]
00010058:   ldr     r3, [r11, #-8]
0001005c:   ldrb    r3, [r3]
00010060:   cmp     r3, #0

Console  Registers  Problems  Executables  Debugger Console ×  Memory
Lab1 Debug [C/C++ Attach to Application] C:\ARM_TOOLCHAIN\bin\arm-none-eabi-gdb.exe (7.5.1)
A program is being debugged already.
Are you sure you want to change the file? (y or n) [answered Y; input not from terminal]
Reading symbols from E:\Master\Embedded-System-Online-Diploma\unit2_C_Programming\codeforces\Lab1\learn-in-depth.elf...done.

si
si
reset () at startup.s:4
4           bl main
```

- Now he displayed current file and you can debug using GUI.

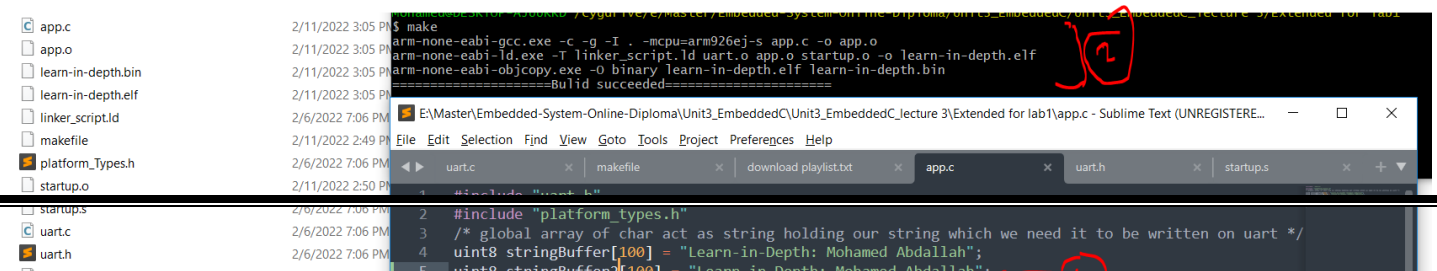
Makefile



1. Clean_all → target, no dependences, role: remove all files .elf, .bin and .o.
2. How to call it? make then write target name, if we didn't write target it will execute first target.
3. startu.o → target, dependences → startup.s, role → assembler.



1. target → startup.o, dependences → startup.s, role → assembler.
2. Target → uart.o, dependences → uart.c, role → compile but not link (-c).
3. Same as 2.
4. Target → learn-in-depth.elf, dependences → startup.o, uart.o, app.o role → link.
5. Target → learn-in-depth.bin, dependences → learn-in-depth.elf, role → obtain img to burn on board (without debugging info).
6. As when we call make, the first target will be executed we put all target and made it depended on .bin so it will generate all missing dependences files.
7. Call make it generated all files and build succeeded as show.



1. We will make some changes on app.c file as shown.
2. Then build again we notice that it build app.c only then link again and generate binary image finally.
   (changed files only will be build and all files depended on that changed files as shown not all file).

```
#@copyright : Mohamed Abdallah
CC=arm-none-eabi-
CFLAGS= -g -mcpu=arm926ej-s
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
AS=$(wildcard *.s)
ASOBJ=$(AS:.s=.o)
PROJECTNAME=learn-in-depth
all: $(PROJECTNAME).bin
    @echo ====================Bulid succeeded=====================

%.o: %.s
    $(CC)as.exe $(CFLAGS) $< -o $@
```

1. %.s → anyfile.s.
2. $< → replaced with dependences file.
3. $@ → replaced with target file.
4. $(INC) -→ replaced with value of the variable.
5. Wildcard *.c → all files.c.
6. SRC:.c=.o → replace .c with .o in files of SRC.